Programmier-Klausur
Programmieren 2 – PZR 2 (30.3.2021)

Erlaubte Hilfsmittel:

- Rechner (auch der eigene)
- IDE (IntelliJ, Netbeans, Eclipse)
- Dokumentationen Java (auch die Online-Javadoc von Oracle)
- Nur bei der Abgabe: Webbrowser um die Lösungen in Moodle einzustellen

Ablauf:

Dauer: 90 Minuten

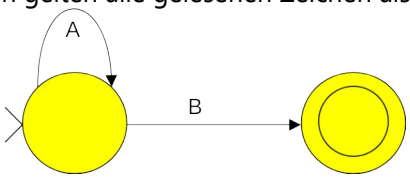
1. Lesen Sie die Aufgabe aufmerksam!
2. Implementieren Sie die Aufgabe!
3. Zippen Sie Ihren Projektordner.
4. Stellen Sie das gezippte File in Moodle ein.

Fragen? Fragen Sie so, dass alle die Frage und die Antwort hören können.

Aufgabe:

Implementieren Sie einen endlichen deterministischen Automaten. Ein solcher Automat startet in einem Startzustand. Dann kann man ihm Zeichen übergeben. Wir nehmen einen String als Datentyp dieser Zeichen. Ein solcher Automat reagiert auf drei mögliche Arten auf ein solches Zeichen: 1) Er verbleibt im gleichen Zustand, 2) er wechselt in einen anderen Zustand oder 3) er dokumentiert einen Fehler, weil das Zeichen in dem aktuellen Zustand nicht erlaubt ist.

Einer oder mehrere Zustände eines solchen Automaten werden als Endzustände definiert. Gelangt der Automat in einen solche Zustand dann gelten alle gelesenen Zeichen als gültiger Satz einer Grammatik.



Das folgende Beispiel zeigt einen endlichen Automaten. Der Startzustand (links) ist mit einem kleinen Haken symbolisiert. Der Endzustand (rechts) ist durch den inneren schwarzen Ring symbolisiert. Zustände, die keinen solchen Haken haben oder keinen inneren Ring wären weitere Zustände des Automaten (nicht in der Abbildung). Dieser endliche Automat würde zum Endzustand gelangen, wenn eine beliebige Anzahl (auch 0) des Symbole A eingegeben werden gefolgt von einem einzelnen B.

Sie sollen einen solchen Automaten implementieren und eine Kleinigkeit mehr: Ein Callback-Objekt. Bleiben wir zunächst aber bei dem Automaten. Das Interface eines Automaten sieht in etwa so aus (das muss nicht vollständig sein):

```
interface StateMachine {
   void readSign(char sign);
}
```

Ein Automat ist also in der Lage, ein Zeichen zu lesen und ändert daraufhin seinen Zustand. Beachten Sie, dass hier jedes Zeichen erlaubt ist, auch die, die nicht im Vokabular des Automaten enthalten sind. Damit muss der Automat geeignet umgehen. Wie? Ihre Entscheidung.

Soweit, so einfach. Kommen wir zurück zum Callback: Sie erinnern sich an den Schritt 4 unseres Spiels. Dort haben wir uns mit dem Prinzip des Callbacks beschäftigt. Das Prinzip ist einfach: Es gibt ein Objekt, das

Ereignisse produzieren kann (in unseren Fall der Automat). Diese Ereignisse sollen in unserem Fall sein:

- Der Automat wurde gestartet und befindet sich nun im Startzustand
- Der Automat wechselt in einen anderen Zustand, der kein Endzustand ist
- Der Automat wechselt in einen Endzustand

Es gibt ein anderes Objekt, das über diese Ereignisse informiert wird. Definieren Sie dafür ein Interface (StateMachineListener), dass für jedes Ereignis eine Methode anbietet. Hinweis: Callbackmethoden liefern meistens keinen Rückgabewert. Das Interface wird etwa so aussehen:

```
interface StateMachineListener {
   // das muss nicht vollständig sein...
   void initialStateReached(Parameter?);
   ....
}
```

Noch einmal: Diese Methoden soll der Automat aufrufen, wenn er eine Zustandsänderung vollzog. Das ist kein komplexes Interface.

Das sollte Ihnen bekannt vorkommen. Das Verhältnis des Automaten zum Listener ist wie die Protokollmaschine zur Spielemaschine in diesem Semester.

Implementieren Sie nun eine Klasse, die sich dieses Interface implementiert. Objekte der Klasse sollen nicht viel tun, sondern sich lediglich die Aufrufe in einer Liste merken. Das ist wirklich sehr sehr wenig Code. Bitte denken Sie hier nicht zu komplex!

Sind Sie damit fertig, schreiben Sie Tests.

```
void testx() {
   // Erzeugen Sie in jedem Test ein Listener-Objekt
   StateMachineListener listener = new YourStateMachineListenerImpl();
   // Erzeugen Sie danach einen endlichen Automaten mit dem Listener
   StateMachine machine = new YourStateMachine(listener);
   // lassen Sie den Automaten Zeichen lesen
   machine.readSign('A');
   // prüfen Sie nun mit dem Listener-Objekt, ob dieser Test korrekt war. Dort sollte ein Reaktion erzeugt
worden sein
}
```

Schreiben Sie so viele Tests wie nötig sind, um alle wesentlichen Eigenschaften des Automaten zu testen. Implementieren Sie danach den Automaten. Alle Prüfungsteilnehmer:innen erhalten einen eigenen Automaten per E-Mail (Ihr HTW-Account). Schauen Sie also in Ihre Mailbox. Wenn Sie dort keinen finden, melden Sie sich bitte. **Viel Erfolg und don't panic.**